BTeV Muon Database Design and Production Monitoring

Eric W. Vaandering
Vanderbilt University

`http://www.hep.vanderbilt.edu/~somewhere`

21 March 2001
Revised: 12 July 2001

Abstract

We describe the design and implementation of the BTeV Muon System production database. As part of our quality assurance and control process, this database is designed to allow us to easily monitor the performance of detector elements and determine correlation. In order to do this we need to understand exactly where each piece of the muon detector comes from and what its properties are.

1 Introduction

We will use a combination of specialized input devices, user driven input programs (mostly GUI) and a database back end to store the information we accumulate during the muon system production and comissioning. The goal is to have each person working on constructing or testing the muon system to easily enter information into the database immediately upon completing their task.

User access to accumulated data is simplified by the existence of database API's in a variety of programming languages.

Throughout the design of this system, modularity has been a driving concern. This allows us to replace parts of the software archeticture without requiring an extensive re-write.

2 Tools and modules

We are using the following toolkits and modules to construct this system:

MySQL Database:
The database is the ultimate repository of all the information which has been collected. The master database currently resides at Vanderbilt. Mirrors may be set up and UPR and UI. See `http://www.mysql.com`

CueCat barcode reader:
This is a promotional barcode reader distributed by Radio Shack. It reads all popular barcode formats, but is a not particularly robust. `http://www.crq.com/master_templ.cfm?view=products&products=cuereader`

ScanData barcode reader software:
This software translates the CueCat output format into the barcode numbers. `http://www.users.qwest.net/~rrapplean/scandata.html`

KDE: We use KDE as a C++ graphical toolkit to develop user applications for entering information into the database. `http://www.kde.org`

MySQL++ toolkit:
A C++ API for accessing MySQL databases `http://www.mysql.com/downloads/api-mysql++.html`

DBI: A Perl API for accessing MySQL (and other) databases

Computer Boards DAS-08/Jr:
A generic ISA DAQ board with digital and analog I/O used in the tension measurement station.

PHP: Nothing has been done here yet, but this is a leading candidate for web-based access to the database.

A couple of these choices should be commented on. MySQL has been chosen as the database for several reasons. First, it is free software licensed under the GPL. It is also very fast and has plenty of readily available documentation. While it is not as complete of an implementation as Oracle, PostgresSQL, etc., the features it is missing are not ones we need. Easy access to the database through a variety of programming language API's is also a major selling point.

The CueCat barcode reader is being used primarily because it is free and software to communicate with the device is widely available. In the future we may wish to move to a more industrial solution from a company like Symbol.

3 Database design

A database is organized into several logical elements:

Field:	Contains a single piece of information
Entry:	A collection of fields, corresponding to one set of values for a single logical element
Table:	A collection of entries for different elements of a particular type
Database:	A collection of related tables

The database design we are using is in the "third normal form." For a longer discussion of what this means and how a database is properly designed see [1]. Essentially it boils down to three requirements:

1. Each field of a database contains only one piece of information
2. Any field which has values in common across multiple entries should use a unique identifier and another table
3. No two fields in a table should depend on eachother

To take a simple example, the field containing a record of which person assembled a piece of equipment should contain a unique identifier, not their name because the name might have been mis-spelled (rule 2). The phone number of the person who assembled something is not a property of the thing, but of the person (rule 3).

The structure of the muon system database is shown in Figure 1. Several additional tables are shown in Figure 2. In both cases the color code is as follows: Tables outlined in blue already exist. Those shown in black are planned. Fields shown in red are indexed for fast look up. Other fields must be searched on which is more time consuming. Each index takes up a significant amount of disk space, so the number of indices should be kept to a minimum.

The "tree"-like symbols indicate the one-to-many relationships within the database. For instance, there can (and will) be many measurements of the efficiency of a single tube. The ID numbers in each entry indicate what other entries they are associated with. In addition, each table has a timestamp which is not shown.

It should also be noted that the structure of the database needn't follow the timeline of the construction and testing processes, but is instead based on which pieces of information are properties of which elements. Information can be added to a table several times during the construction process.

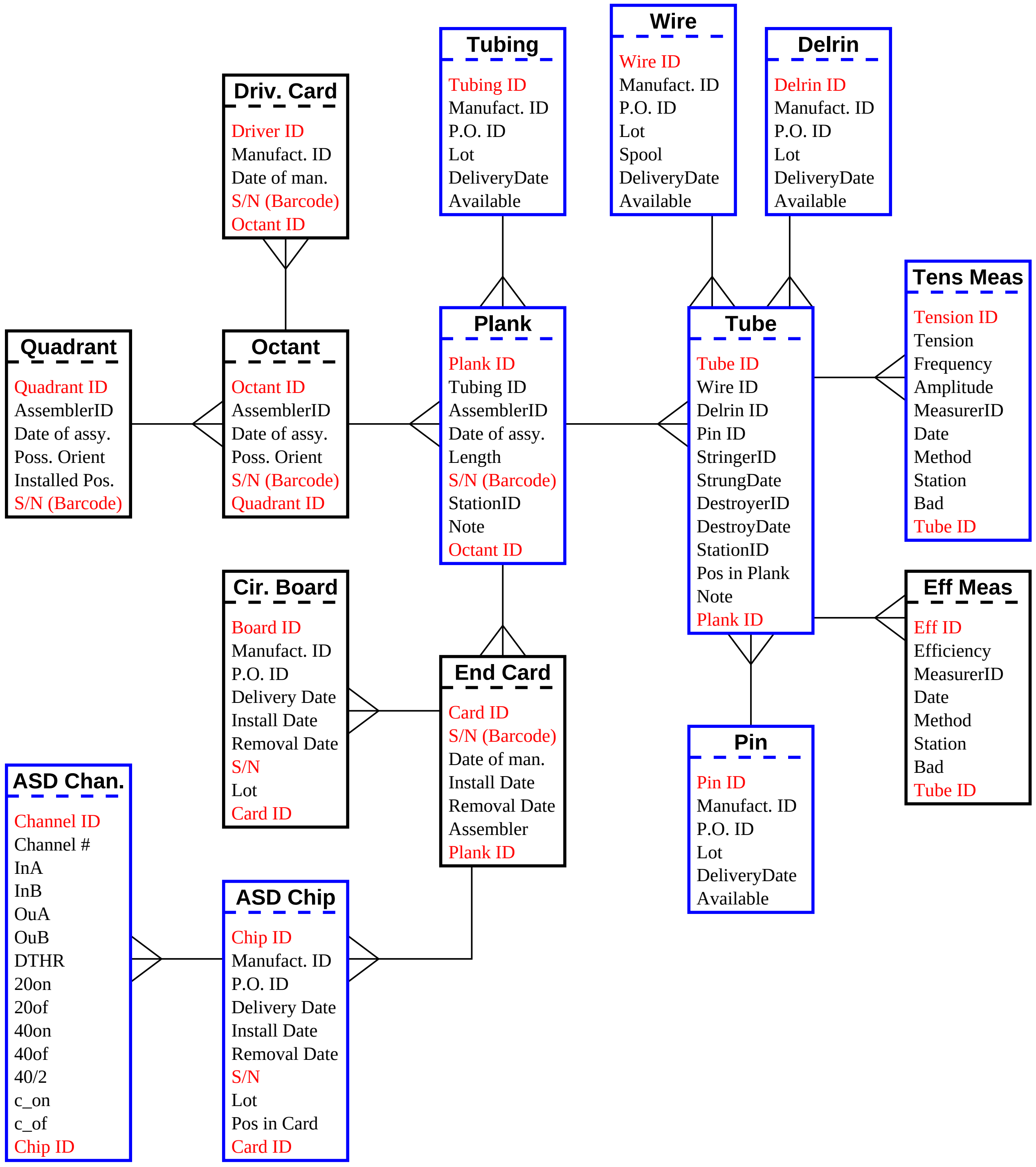

Figure 1: The database model for the BTeV muon system and the inter-relationships between tables

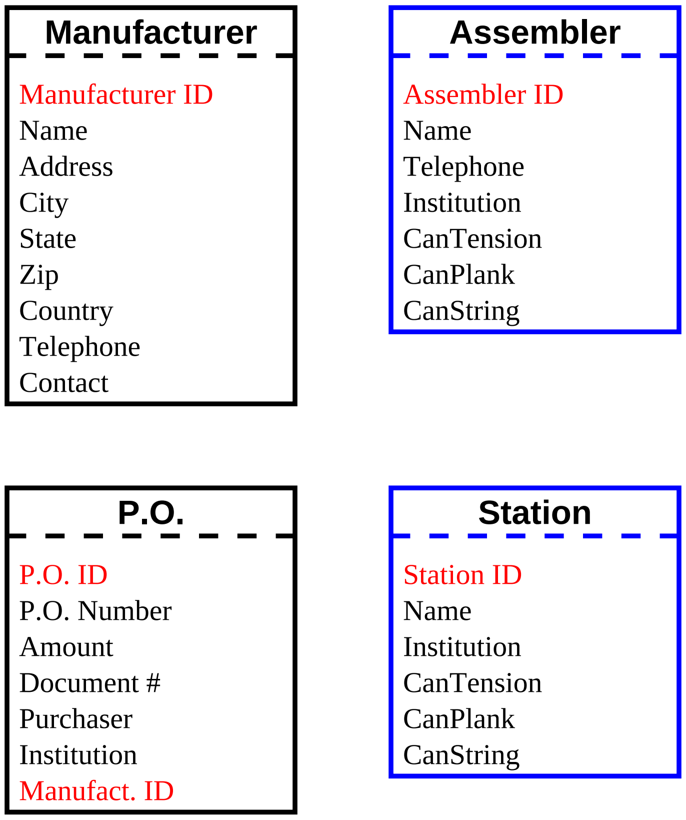

Figure 2: Generic tables used in the BTeV muon system database.

4 Summary

We have designed and partially implemented what we believe is a close-to-final database design to be used during the construction and testing phase of the muon system. Additional tables will be implemented as data to fill them becomes available. Adding additional data to a table is also fairly trivial (assuming that acquiring the data is easy).

We've also determined that using barcodes to track the assembly of detector components will be a useful timesaving and accounting technique.

References

[1] R. J. Yarger, G. Reese, and T. King, *MySQL & mSQL*, O'Reilly & Associates, Inc., 1999.